# Practical Machine Learning Course Project

Douglas Wirtz

May 5, 2016

---

## Introduction

This is the final project for the Practical Machine Learning course in the data science specialization offered by Johns Hopkins University on Coursera.The simple goal of this project is to create a prediction model from a set of data and use it to predict 20 different test cases.

The data are collected from large amounts of personal activity devices such as *Jawbone Up*, *Nike FuelBand*, and *Fitbit*. These types of devices quantify self movement which allow people to improve their health or find patterns in their behavior. Oftentimes, users of these devices will use them to quantify how much of a particular activity they do, but rarely use them to quantify how well they do it. This project will focus on the data collected from the accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perfom barbell lifts correctly and incorrectly in 5 different ways. From this, a model is created to predict the manner in which they did the exercise.

### Data

For more information on the data set, you can find it here under the section on the Weight Lifting Exercise Dataset.

The training data for model creation are available here.

The test data are available here.

---

## Get the Data and Setup for Analysis

To begin, the following pacakages were loaded and a seed was set for reproducibility purposes.

```r
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(ggplot2)
library(lattice)
library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

set.seed(18)
```

Download the data if they don't already exist.

```r
trainCSV <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
testCSV <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
if(!file.exists("train.csv")){
    download.file(trainCSV, destfile = "train.csv")
}
if(!file.exists("test.csv")){
    download.file(testCSV, destfile = "test.csv")
}
train <- read.csv("train.csv", na.strings = c("NA", "#DIV/0", ""))
test <- read.csv("test.csv", na.strings = c("NA", "#DIV/0", ""))
```

To estimate the out-of-sample error, the train data is split into a smaller training set `train1` and a validation set `train2` based on the "classe" variable.

```r
inTrain <- createDataPartition(train$classe, p = 0.7, list = FALSE)
train1 <- train[inTrain, ]
train2 <- train[-inTrain, ]
dim(train1); dim(train2)

## [1] 13737   160

## [1] 5885  160
```

Now the data must be cleaned by removing the variables with nearly zero variance, removing the variables that are almost always "NA", and removing the variables that don't make sense for prediction.

```r
## Remove the variables with nearly zero variance
nzv1 <- nearZeroVar(train1)
train1 <- train1[, -nzv1]
train2 <- train2[, -nzv1]
```

```
nzv2 <- nearZeroVar(train)
train <- train[, -nzv2]
test <- test[, -nzv2]
dim(train1); dim(train2); dim(train); dim(test)

## [1] 13737   125

## [1] 5885  125

## [1] 19622   117

## [1]  20 117

## Remove the variables that have "NA" greater than or equal to 95% of the
time
trainNA1 <- sapply(train1, function(x) mean(is.na(x))) >= 0.95
train1 <- train1[, trainNA1 == FALSE]
train2 <- train2[, trainNA1 == FALSE]

trainNA2 <- sapply(train, function(x) mean(is.na(x))) >= 0.95
train <- train[, trainNA2 == FALSE]
test <- test[, trainNA2 == FALSE]
dim(train1); dim(train2); dim(train); dim(test)

## [1] 13737    59

## [1] 5885   59

## [1] 19622    59

## [1] 20 59

## Remove the variables that don't make sense for prediction
colnames(train1)

##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "num_window"
##  [7] "roll_belt"            "pitch_belt"           "yaw_belt"
## [10] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
## [13] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [16] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [19] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [22] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [25] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [28] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [31] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [34] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [37] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [40] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [43] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [46] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [49] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [52] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [55] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [58] "magnet_forearm_z"     "classe"
```

Based on what we see above, the first 5 columns will be removed because they don't provide any help for model prediction.

```
# Remove first 5 columns
train1 <- train1[, -(1:5)]
train2 <- train2[, -(1:5)]
train <- train[, -(1:5)]
test <- test[, -(1:5)]
dim(train1); dim(train2); dim(train); dim(test)

## [1] 13737    54

## [1] 5885   54

## [1] 19622    54

## [1] 20 54
```

---

## Build the Model

To build the model, I will start with a random forests method. If I don't think the model is good enough, I will try a different method. I will fit the model on `train1` while using a 3-fold cross-validation for the training control.

```
# Fit model using training partition (train1) and random forest method
trainControl <- trainControl(method = "cv",number = 3)
fit <- train(classe ~ ., data = train1, method = "rf", trControl =
trainControl)
fit$finalModel

##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    5 2651    2    0    0 0.0026335591
## C    0    5 2391    0    0 0.0020868114
## D    0    0    7 2244    1 0.0035523979
## E    0    0    0    5 2520 0.0019801980
```

As you can see, the random forest generated 500 different trees trying 27 variables at each split. The resulting estimate of the error rate is 0.2% so this appears to be a very accurate model. The next step is to continue with this model and evaluate it against the validation set.

---

## Evaluate the Model

In evaluating the model, I will use the fitted model to predict the `classe` variable in `train2`. This is shown using a confusion matrix.

```
# Predict on validation data (train2)
prediction <- predict(fit, newdata = train2)
confusionMatrix(train2$classe, prediction)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    1 1135    2    1    0
##          C    0    3 1023    0    0
##          D    0    0    1  962    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9992)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9974   0.9971   0.9979   0.9991
## Specificity            1.0000   0.9992   0.9994   0.9996   0.9998
## Pos Pred Value         1.0000   0.9965   0.9971   0.9979   0.9991
## Neg Pred Value         0.9998   0.9994   0.9994   0.9996   0.9998
## Prevalence             0.2846   0.1934   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1738   0.1635   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9997   0.9983   0.9982   0.9988   0.9994
```

The accuracy for this prediction is 99.8% which is very good. This also means the out-of-sample error is very small at 0.2%. I consider this random forest fit to be good enough to predict on the test data.

---

## Re-fit Model and Make Test Predictions

Before I predict on the test data, I will re-fit the model using the entire training data using the same parameters as above.

```
# Re-fit model using entire training data
fit2 <- train(classe ~ ., data = train, method = "rf", trControl =
trainControl)

# Predict on test data
prediction2 <- predict(fit2, newdata = test)

# Write predictions to a file
prediction2 <- as.character(prediction2)
to_file <- function(x){
    n <- length(x)
    for(i in 1:n){
        filename <- paste0("num_", i, ".txt")
        write.table(x[i], file = filename, quote = FALSE, row.names = FALSE,
                    col.names = FALSE)
    }
}
to_file(prediction2)
```